

(12) UK Patent Application (19) GB (11) 2 329 812 (13) A

(43) Date of A Publication 31.03.1999

(21) Application No 9820543.8

(22) Date of Filing 21.09.1998

(30) Priority Data

(31) 08936275 (32) 24.09.1997 (33) US

(71) Applicant(s)

Sony Pictures Entertainment Inc.
(Incorporated in USA - Delaware)
10202 West Washington Boulevard, Culver City,
California 90232, United States of America

(72) Inventor(s)

James Merce
Richard J Oliver
Jeffrey Mark Claar
Roger M Duvall

(74) Agent and/or Address for Service

D Young & Co
21 New Fetter Lane, LONDON, EC4A 1DA,
United Kingdom

(51) INT CL⁶

H04N 5/262

(52) UK CL (Edition Q)

H4T TBAX T109 T150 T151

(56) Documents Cited

GB 2323738 A GB 2096868 A

(58) Field of Search

UK CL (Edition P) H4T TBAX TBLA TBLC TBLM TBLX
INT CL⁶ H04N 5/222 5/262 5/265 5/268
ONLINE: WPI, INTERNET

(54) Abstract Title

Configurable Graphic User Interface for audio/video data manipulation

(57) A configurable graphic user interface (GUI) 705 for audio/video (A/V) data is associated with a data descriptor engine 710 coupled to the A/V file system 715 containing the A/V data. A type of user interfacing to the GUI 705 is identified. A display generator 720 accesses the A/V file system 715 and generates a display of the A/V data dependent upon the type of user. A menu constructor 725 generates user selectable options/features to be displayed in the GUI 705, the options/features displayed being dependent upon the type of user. A manipulation engine 730 receives user input and determines operations to be performed, the determination being based upon the type of user.

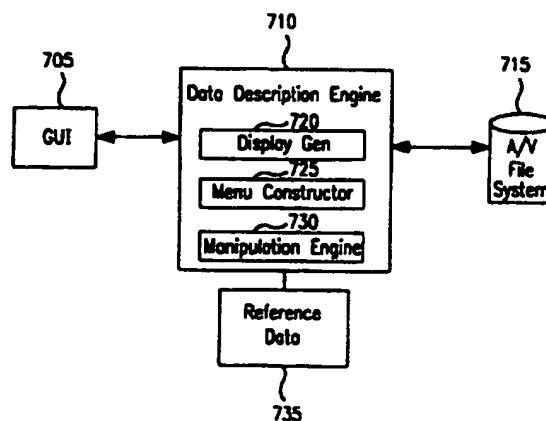


FIG. 7

USER INTERFACE SYSTEMS AND METHODS

The present invention relates to the field of audio/video editing. Specifically, the present invention relates to user interface systems and methods, such as those suitable for editing digitized audio/video data across a network.

Currently, audio/video editing is most frequently performed according to the system illustrated in Figure 1. An editing workstation today includes local disks on which audio/video is stored. Editing may be performed via a display screen, but is limited to the locally stored audio/video data. This type of a system provides insufficient functionality for editing a large amount of audio/video data stored on multiple workstations.

More specifically, this system is insufficient because current editing workstations do not perform editing across the boundaries of workstations, over networks. Audio/video data on each workstation must be edited separately, and the output of the editing may then be input into a mixing console to create a final audio/video output. If there are any problems with the final audio/video output, then the tedious process of editing the various audio/video data tracks residing on each separate workstation has to be repeated. This process continues until the final output is satisfactory.

The present invention provides a system and method for providing a configurable user interface to audio/video (A/V) data. In one embodiment, a data descriptor engine is coupled to the A/V file system containing the A/V data. A type of user interfacing to the GUI is identified. A display generator
5 accesses the A/V file system and generates a display of the A/V data dependent upon the type of user. A menu constructor generates user selectable options/features to be displayed in the GUI, the options/features displayed dependent upon the type of user. A manipulation engine receives
10 user input and determines operations to be performed, the determination based upon the type of user.

The invention will now be described by way of example with reference to the accompanying drawings, throughout which like parts are referred to by like references, and in which:

Figure 1 illustrates a prior art audio editing method;

5 Figure 2 illustrates an overview of one embodiment of the present invention;

Figure 3 illustrates in further detail the software modules according to one embodiment of the present invention;

10 Figure 4 illustrates a data structure defining a track according to one embodiment of the present invention;

Figures 5A-E illustrate various editing operations performed on a track of audio data;

Figure 6 illustrates an example of the graphical user interface that may be used to perform editing;

15 Figure 7 illustrates one embodiment of a system that generates user configurable interfaces in accordance with the teachings of the present invention;

Figure 8 illustrates one example of a graphical user interface for a different type of user; and

Figure 9 illustrates an alternate example of a graphical user interface for a particular user.

The present invention is a method and apparatus for generating a user configurable interface. In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent to one of ordinary skill in the art that these specific details need not be used to practice the present invention. In other instances, well-known structures, interfaces, and processes have not been shown in detail in order not to unnecessarily obscure the present invention.

Figure 2 illustrates an overview of one embodiment of the present invention. Although the following sections describe an audio editing process, the same technique may be successfully applied to a video editing process. As illustrated, the system according to the present invention comprises at least one editing station coupled to a plurality of remote stations (remote stations 1-N). Remote stations 1-N contain audio/video data stored in digitized form on one or more storage disks. According to this embodiment of the present invention, a sound engineer may edit the digitized data residing on any of remote stations 1-N from a single location, namely editing station 200. The sound engineer may also edit the digitized data across remote stations 1-N, e.g. by copying audio from a track on one station and pasting the cut audio onto a track on another station. Editing operations across remote stations are described in further detail below.

Editing station 200 may be a personal computer (PC) or any other type of system capable of displaying a graphical application, accepting input via the graphical application and communicating the input to a remote location.

Remote stations 1-N may also comprise a PC or a proprietary audio/video storage/editing station such as a Sony® Corporation player/recorder system. Remote stations 1-N may include slots for multiple removable disks, such as Iomega Jaz disks, that increase the storage available to the remote station and
5 allow the stored data to be easily moved from one remote location to another.

Figure 3 illustrates the software modules according to one embodiment of the present invention. Editing station 200 includes editing application 302, dynamic link libraries (DLLs) 304, and a Remote Procedure
10 Call (RPC) mechanism 306. Editing application 302 allows the sound engineer to perform editing functions via a graphical user interface (GUI). RPC mechanism 306 provides a communication link between the editing station and the remote stations 1-N (from Figure 2). According to one embodiment of the present invention, RPC mechanism 306 is a Microsoft®
15 Windows NT based RPC mechanism. Alternate RPC mechanisms or other network mechanisms for remote communication may also be utilized.

Figure 3 also illustrates an example of the software modules on an exemplary remote station according to one embodiment of the present invention. The exemplary remote station comprises RPC mechanism 308
20 (corresponding to RPC mechanism 306 on editing station 300), DLLs 310 and a driver mechanism 312 that interacts with Digital Signal Processing (DSP) system 314. A sound engineer interacts with remote stations 1-N from editing station 200 via a GUI, the functions specified via the GUI are communicated between editing station 200 and remote stations 1-N via RPC
25 mechanism 306 and 308, and DLLs 310 and DSP system 314 perform all necessary processing to implement the specified functions.

Figure 4 illustrates the data structure of an audio track on an exemplary remote station according to one embodiment of the present invention. An audio track is a static representation of audio that specifies the location of audio data on a disk, the organizational data necessary to correctly playback the audio data as well as a variety of other parameters associated with the audio data. According to one embodiment of the present invention, the various parameters associated with audio data are each specified in a list.

The various lists illustrated in Figure 4 therefore correspond to a single audio track. This track is represented by a data structure that links all the lists together in a logical fashion. By using a single data structure to link all the lists together, one embodiment of the present invention allows all data associated with a particular track of audio to be edited collectively by a common editing interface. Thus, for example, if a sound engineer decides to edit a portion of an audio track, all lists associated with that portion of the audio track may be edited simultaneously, unlike prior art systems in which various aspects of the track may have to be edited separately.

The mechanism that performs edits across the track data structure of Figure 4 involves several components. According to one embodiment, the components are implemented as C++ classes. The primary components are: Tracks, EventLists and Events. Each of these components is illustrated in Figure 4, and described in further detail below. Although the following description assumes a C++ implementation, the same or similar functionality may be implemented in other programming environments.

A "track" is a public object that is used by audio applications to

represent static streams of audio and other time based data. A track is implemented as a named object (i.e. has a string name) that maintains a collection of event lists (described below). A track exposes an edit interface that is applied to each event list. According to one embodiment, a track
 5 maintains at least one event list, namely the SoundList (see Figure 4).

"Event lists" are doubly linked data structures that maintain two basic types of events. The default event is termed a "silence" event and the other event is termed a "non-silence" event. There is a separate event list for each type of silence/non-silence event combination. For example, as illustrated in
 10 Figure 4, solid color-filled circles along the time of the audio represent the occurrence of sound while unfilled circles represent silence. Other lists are also contemplated. For example, as shown in Figure 4, the gain list illustrates how much gain is applied to particular portions of an audio track. For example, the solid filled circle indicates that a certain scalar value of gain
 15 is to be applied while the unfilled circle represents that unity gain is to be applied. Figure 4 also shows "other lists" which, as is now apparent to one skilled in the art, can include a variety of parameters including parameters related to video or other editing operations. Each type of event maintains at minimum a length measured in samples and other data needed to define the
 20 event behavior. According to one embodiment, virtually all of the edit behavior of event lists is implemented in a C++ class.

Each type of event maintains two important virtual functions: SlideStart(...) and SlideEnd(...). When new non-silence events are created, each event must reimplement these functions. This is due to the fact that
 25 when a start and/or end time of an edit operation does not match with the beginning of an event, a "seam" must be created. This is accomplished by

copying the appropriate event, "sliding" the end of the source event and then "sliding" the start of the copied event. This sliding mechanism is illustrated in further detail below (see Figure 5C below).

Event lists are time based data structures that contain linked events
 5 where each event maintains at minimum a length measured in samples. The sum of the lengths of all of the events plus the start offset (explained in further detail below) equals the length of the event list. By default, the first event starts at the absolute time of zero samples unless a start offset is present, in which case the first event starts at the value of the start offset.
 10 The absolute time (measured in samples) at the start of an event is measured by adding up the lengths of all of the events previous to it including the start offset.

For the purposes of this discussion, tracks and event lists are used interchangeably. Tracks represent multiple event lists. A working event list
 15 is a list that contains the events being built to model a specific type of audio behavior. Working events lists are initially created with a very large silence event. Once the working event list has been created, edit operations can be applied to it. Single events can be added to the list by calling an appropriate Add(...) function. In addition, according to one embodiment of the present
 20 invention, six primitive edit operations can be applied. They are:

Clear()
 Cut()
 Copy()
 25 Paste()
 Insert()
 Insert() (space)

Tracks, event lists and any other objects that model editing use the

above terms when defining an edit interface. The edit operations as they apply to event lists are discussed below and illustrated in Figures 5A-E. According to an alternate embodiment of the present invention, other edit operations may also be performed.

5 "Clear" replaces a section of events with silence without affecting the times of the events after the end time. Clear is effectively an overwrite operation. The clear edit operation is illustrated in Figure 5A. "Paste" operations pastes a source list into a destination list. The start and end times of the paste are determined by the source list. Paste operations overwrite,
10 hence the times of events after the end time remain unchanged, as illustrated in Figure 5B.

 "Insert" operations insert a source list into a destination list. The point of insert is determined by the start time of the source list. The times of all events at and after the point of insert are shifted by the length of the
15 source list. This is shown below in Figure 5C. "Insert space" simply inserts a blank event list of a specified length instead of a source list into a destination. "Cut" removes a section of events from an event list. All of the events after the end time are shifted earlier in time by the length of the cut piece, as illustrated in Figure 5D. Finally, "Copy" operations copy a section of an
20 event list while the source list remains unaffected, as illustrated in Figure 5E.

 Figure 6 illustrates an examples of a GUI screen that may be generated by editing application 302 at editing station 200. This GUI screens allows the sound engineer to specify a range for editing functions to be performed on the audio data. For example, the screen in Figure 6 illustrates a display of
25 seven separate tracks (FXL, FXC, FXR, FXS, FoleyL, FoleyC and FoleyR).

These seven tracks may reside on a single station or on separate stations on a network. This ability for a sound engineer to access multiple stations across a network for simultaneous editing from a single remote location represents a significant advancement over the prior art methods of audio/video editing.

5 According to one embodiment of the present invention, the sound engineer can perform multiple edit operations across as many remote stations as he or she desires, thus giving the sound engineer the flexibility to edit a large amount of audio very simply. Thus, for example, a sound engineer may perform any of the edit operations described above to the highlighted portion
10 of the screen. The highlighted portion corresponds to a portion of audio data on multiple tracks of audio data, where each track comprises multiple lists.

The visual representation of the audio allows the sound engineer great flexibility to specify edit functions. For example, the sound engineer may highlight a predetermined portion of the screen and specify edit
15 functions to be performed on the entire highlighted area. This highlighted area, as illustrated in Figure 6, may include both shaded and unshaded portions of the screen. The shaded portions indicate the active audio portions (non-silence) of each track, while the white spaces indicate silence. When the sound engineer specifies an edit function, the function is applied
20 across the tracks, thus allowing the user to edit the multiple tracks simultaneously. More significantly, these tracks may reside on separate machines across the network, in contrast with the prior art where an editing station could only edit data that resided on a local station.

This ability to perform editing of digitized audio/video data from a
25 single editing station across a network provides significant practical advantages other than simply making the process more efficient. In a

current film editing facility, for example, various editing studios may each contain multiple editing stations to edit different portions of audio/visual data, where the output of all the editing stations is input into a mixing console. In the event a machine is reassigned to a different studio, the entire
5 editing station must be physically moved to the new location. In contrast, according to one embodiment of the present invention, the physical remote stations do not have to be moved. Instead, the remote stations may reside at any location, and each studio may simply be assigned a predetermined set of remote stations. In this manner, the number of remote stations assigned to a
10 studio may change on a daily or even hourly basis without significant problems.

In an alternate embodiment, the data structure can be utilized to support of a variety of GUIs that are customized for different types of users. For example, in the movie industry, different individuals work on
15 production sound, music composition, music editing and sound editing. The work performed by the different individuals involves access, all or in part, to the data structure, an example of which is illustrated by Figure 4. However, these different individuals are accustomed to different user interfaces. For example, production sound or music composition people
20 works with bars, measures and beats to identify the particular portions an operation relates to. Sound editors, including mixers, work with SMPTE code. Similarly, if audio for non-video applications was being processed in this system, the different portions of the audio may be identified by a MIDI code. In this alternate embodiment, the system further includes a data
25 description engine as illustrated in Figure 7. Referring to Figure 7, the GUI 705 containing the A/V data is created by the data description engine 710 accessing the A/V database 715. The data description engine 710 may be

implemented in software to run on existing processors or other processors. Alternately, it may be implemented in a dedicated system of hardware or software. The GUI generated is dependent upon the type of user. Thus, for example, the user may initially be prompted by the data description engine to
5 identify the type of GUI desired, e.g., sound editing, production sound, etc. Alternately, the system may implement a default type of GUI based upon the log-on or some other user identification.

The data description engine accesses the A/V database 715 and the display generator displays the data of interest in a representation configured
10 to the type of user. This is graphically represented by Figures 6 and 8. As noted earlier, Figure 6 is one GUI representation configured for a sound editor. Figure 8 is a GUI representation for a dubber. It should be noted that the data from the different tracks are represented graphically in a different manner. For example, with reference to Figure 6, the sound editor is
15 provided a graphical representation of the audio data (e.g., area 655) in addition to providing the identification of the particular audio currently accessed and represented in each of the tracks shown. The dubber display is also provided the identification 805 of the audio data accessed; however, as the editing operations are not performed at a dubbing station, the graphical
20 representation of audio data is not provided. Similarly, dubbing frequently utilizes offsets to offset different tracks relative to each other (e.g. for mixing); however offsets are typically not used in the editing process. Thus, offset fields are provided (e.g., 810) in the GUI of Figure 8 and not the GUI of Figure 6.

25 The representations may be pre-generated and stored, for example in the reference data file 735 or some other memory or storage media. The representation may be stored as descriptive data or executable code (e.g., java

script or the like). It should be noted that there may be some commonality of representation. For example, the time codes representing in points, out points etc. (e.g., 686-690 and 830-836) may be the same, depending upon the type of user.

5 In addition, the menu constructor 725 (Figure 7) generates the user selectable features/options/operations for the particular type of GUI. As with the data representations, the menus may be pre-generated and stored, for example in the reference data file 735 or some other memory or storage media. The representation may be stored as descriptive data or executable
10 code (e.g., java script or the like). Examples of the different user selectable/features/options/operations can be seen with reference again to Figures 6 and 8. In the dubber GUI, several options, such as "s" safety 812, "R" record 814, "I" input 816, "S" Solo 818 and "M" mute 820. These are implemented as buttons, such that when selected by the user, for example, by
15 placing the cursor over the area of the button and "clicking" the mouse/trackball, the function is enabled. For example, the safety button 812 implements a write protect mechanism on the entire track. The record button 814 immediately initiates recording of the input on the selected track. The input button 816 enables the user to hear what is at the input (as
20 opposed to what is at the output). The mute button 820 mutes the selected track and the solo button 818 mutes all the tracks except the selected track.

 In the editing GUI of Figure 6, similar functions are provided, but are some are implemented differently to accommodate how different users operate. As shown, buttons are provide for the solo 678, mute 680, input 682
25 and record 684 functions. However, unlike a dubbing application, in an editing application implements a tape based multitrack model of recording. Thus, if a user selects the record function, the button will blink to indicate

- waiting to initiate recording and will change to a solid color to indicate when recording is initiated as controlled by the user selecting a master record button (e.g., button 691) on the display. Once the master record is selected, additional track record buttons can be selected and immediately enabled.
- 5 Furthermore, if the master record is again selected to turn off the master recording function, the track record buttons will again blink to indicate a waiting to record state.

Identical functions are also provided such as the tape control functions, e.g., fast forward, reverse, play, stop etc. (690 and 840) and scrolling
10 according to time code (692 and 842). In addition, some functions provided are only found in particular types of GUIs. For example, the programmable track selection buttons 844 and "all" 845 or "none" 846 track selection buttons are very useful in the dubbing application wherein the user is manipulating many different tracks at one time. Furthermore, graphically
15 the functions may be represented differently as defined by the menu constructor. For example, the gain value is represented as a slider icon in the editing GUI and as a numerical value in the dubbing GUI. Selection of the gain column or icon in the appropriate GUI will bring up the gain pop up window as shown in Figure 9 which enables the user to adjust the gain using
20 a graphical slider. It should be noted that Figure 9 all illustrates an alternate embodiment of a dubber GUI which is user configurable. In this illustration, tracks separately identifying input and output are utilized to simplify the information displayed. Other embodiments are also contemplated.

Once the data and user selectable functions are displayed, the user is
25 able to provide input to the system to perform certain tasks. The user input is preferably processed through the manipulation engine (730, Figure 7) which interprets the input and generates instructions understood by the

system. The advantage is that the user can use the particular nomenclature or "lingo" typically used and the system will understand what needs to be done and how to perform the task. Thus, some functions are unique to a particular type of user and some are common although referred to differently. For example, in sound production one operation is referred to as a "swap", in music editing, the same operation is referred to as "verse" and in sound editing the same operation is referred to as a "take". All three refer to the operation of time stamping from a determined point to record multiple versions of the same sound. Thus the operation of recording the data will be the same although referred to differently by different types of users. The manipulation engine can be implemented a variety of ways. In one embodiment, a dictionary is constructed and stored, for example, on storage device 735 to contain possible user functions and corresponding internal functions. The manipulation engine can then search the dictionary for the particular function to determine the internal function to be executed.

As can be seen, the present illustration can be expanded beyond the audio examples provided into video and to other types of audio or video users. Thus it is contemplated that the same system can support a wide variety of users without having to provide specialized user hardware for a particular type of user.

Thus, a method and apparatus for generating a user configurable graphical user interface is disclosed. These specific arrangements and methods described herein are merely illustrative of the principles of the present invention. Numerous modifications in form and detail may be made by those of ordinary skill in the art without departing from the scope of the present invention. Although this invention has been shown in relation

to a particular preferred embodiment, it should not be considered so limited. Rather, the present invention is limited only by the scope of the appended claims.

CLAIMS

1. A system comprising:
a graphical user interface (GUI);
an audio/video (A/V) file system comprising A/V data ; and
a data description engine coupled between the GUI and the A/V file system, said data description engine comprising:
a display generator configured to provide A/V data in a form for display in the GUI for a particular type of user,

a menu constructor configured to generate user selectable options/features for display in the GUI, the user selectable options/features generated dependent upon the particular type of user,
and

a manipulation engine configured to receive user input and determine an operation to be performed, the determination of the operation to be performed dependent upon the particular type of user.

2. A system substantially as herein described with reference to and as illustrated in Fig. 2 to 9 of the accompanying drawings.



The Patent Office

19

Application No: GB 9820543.8
Claims searched: All

Examiner: R F King
Date of search: 26 November 1998

Patents Act 1977 Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK CI (Ed.P): H4T[TBAX, TBLA, TBLC, TBLM, TBLX]

Int CI (Ed.6): H04N 5/222, 5/262, 5/265, 5/268

Other: ONLINE: WPI, INTERNET

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
XE	GB 2 323 738 A [Philips] See whole doc.	1
X	GB 2 096 868 A [Ampex] See whole doc.	1

X Document indicating lack of novelty or inventive step
Y Document indicating lack of inventive step if combined with one or more other documents of same category.

& Member of the same patent family

A Document indicating technological background and/or state of the art.
P Document published on or after the declared priority date but before the filing date of this invention.

E Patent document published on or after, but with priority date earlier than, the filing date of this application.

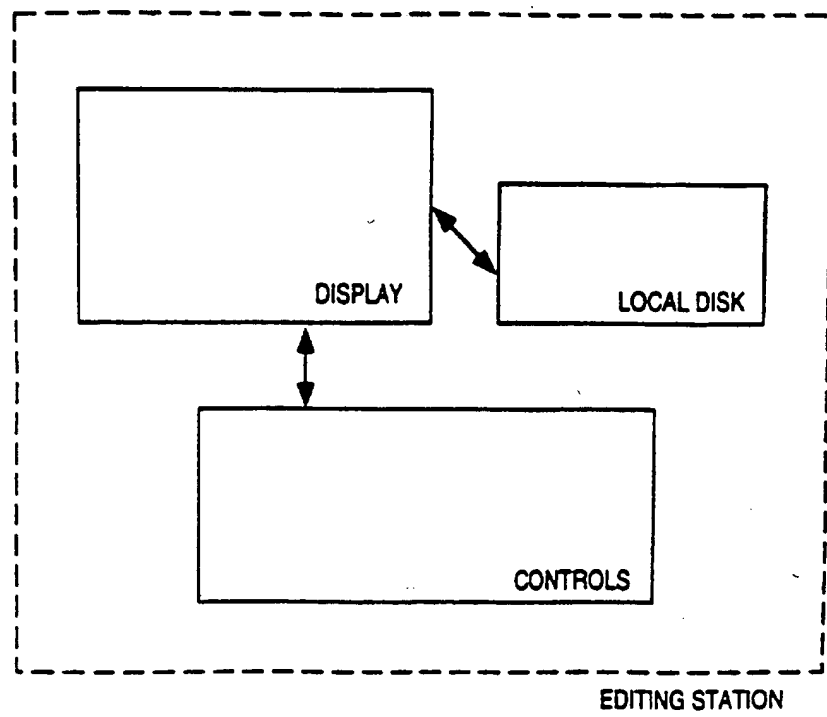


FIG. 1

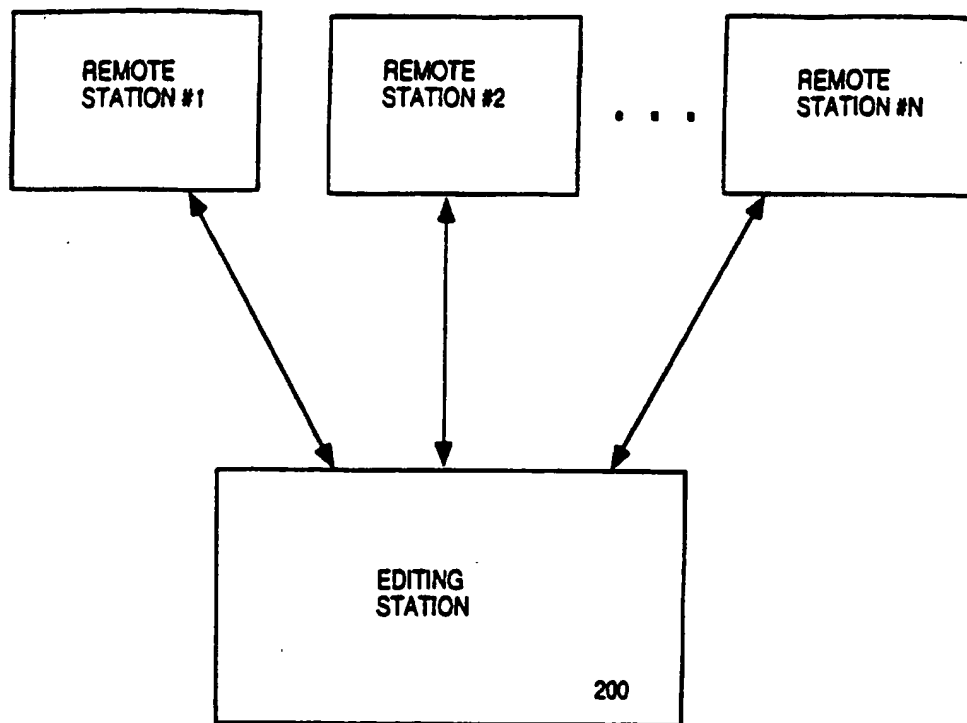


FIG. 2

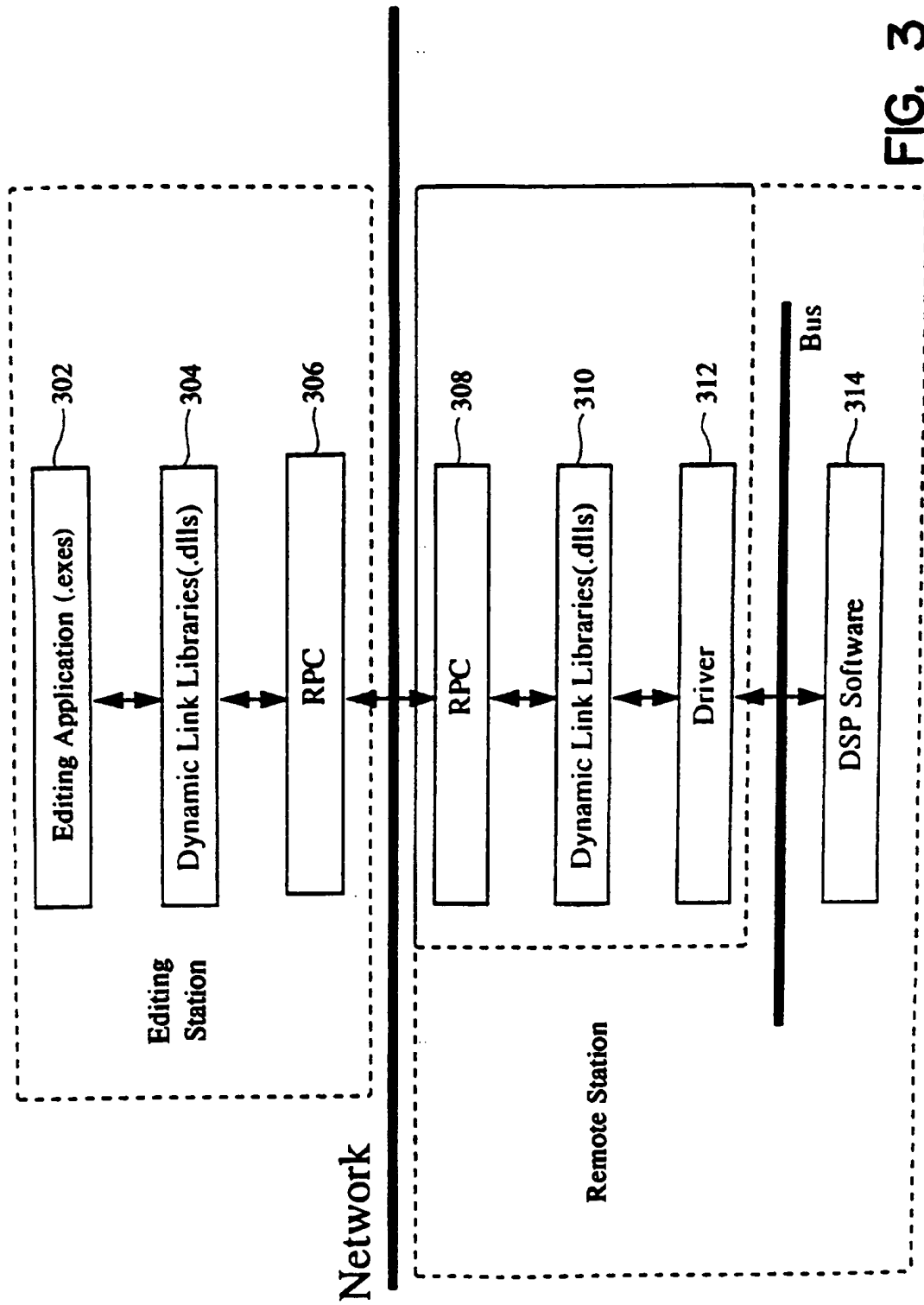


FIG. 3

Tracks Definition

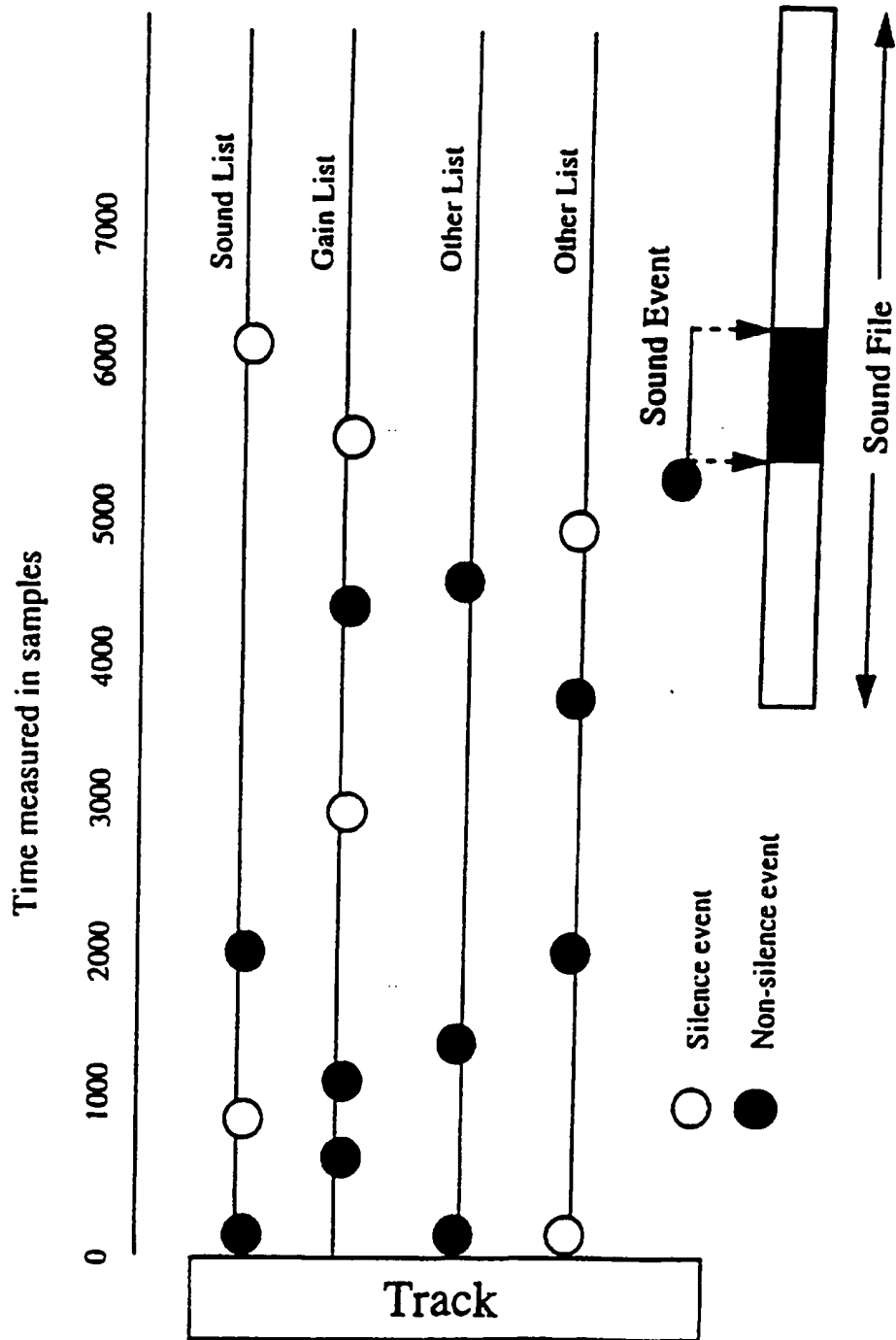
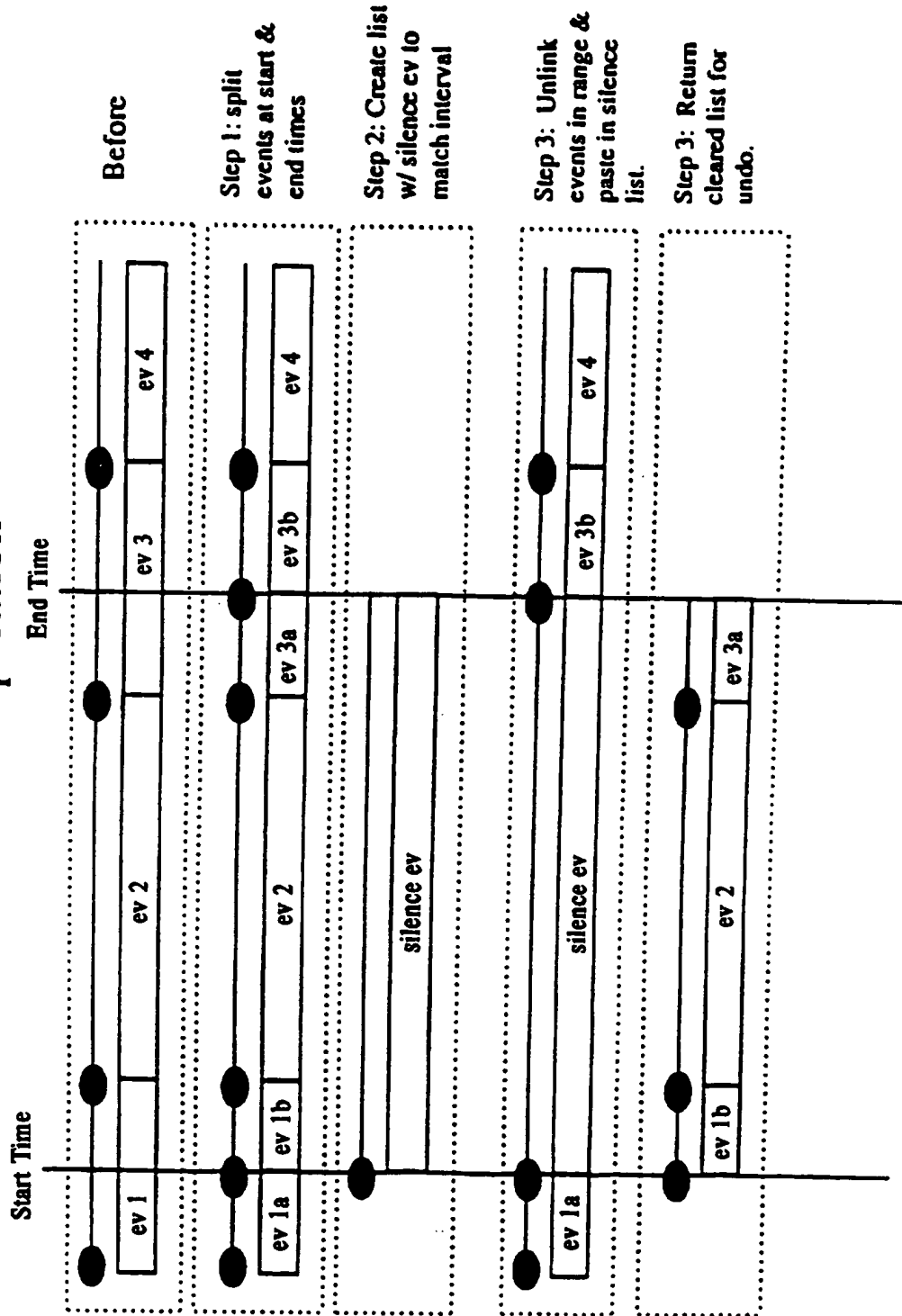


FIG. 4

Clear Edit Operation



Paste Edit Operation

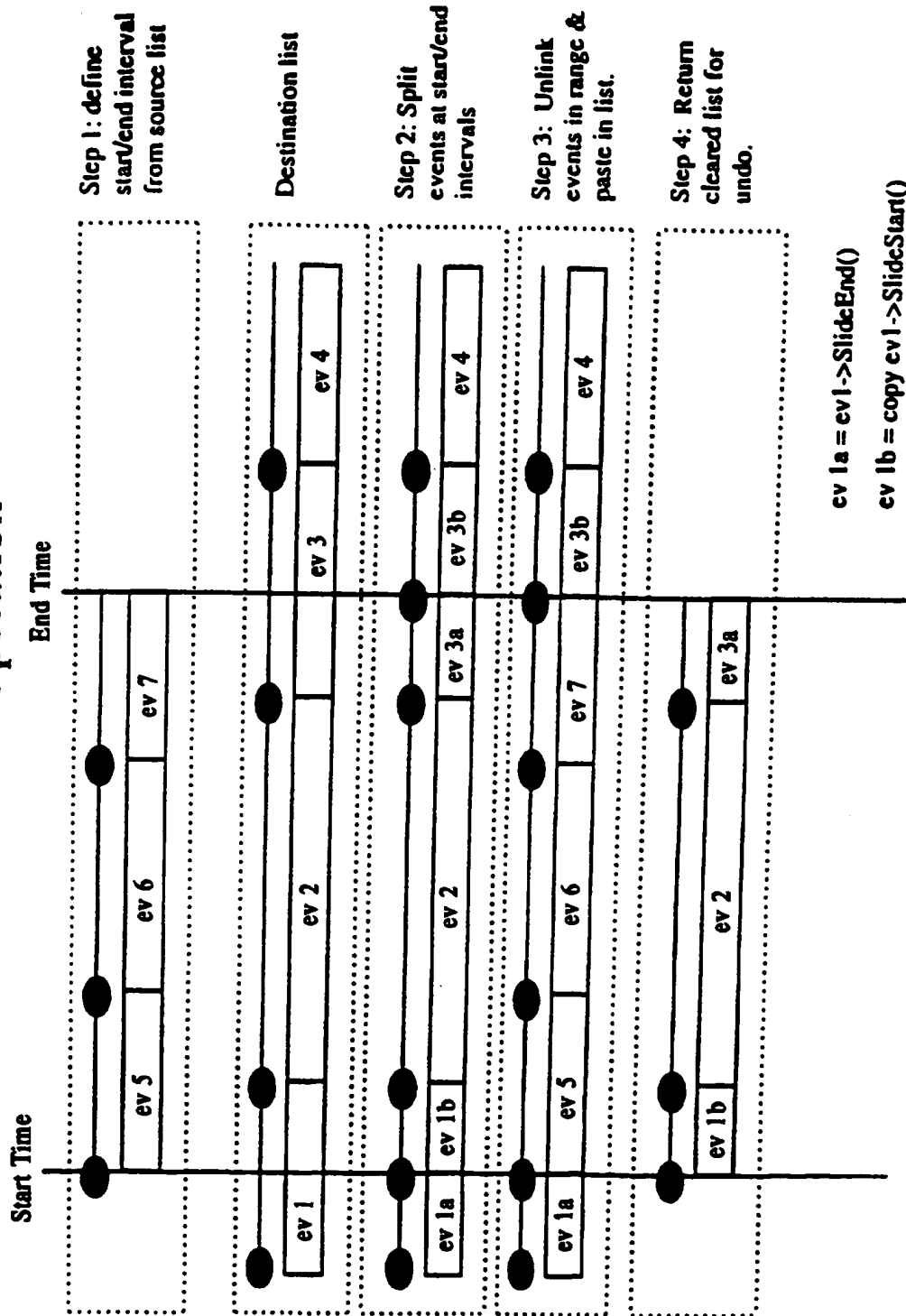


FIG. 5B

Insert Edit Operation

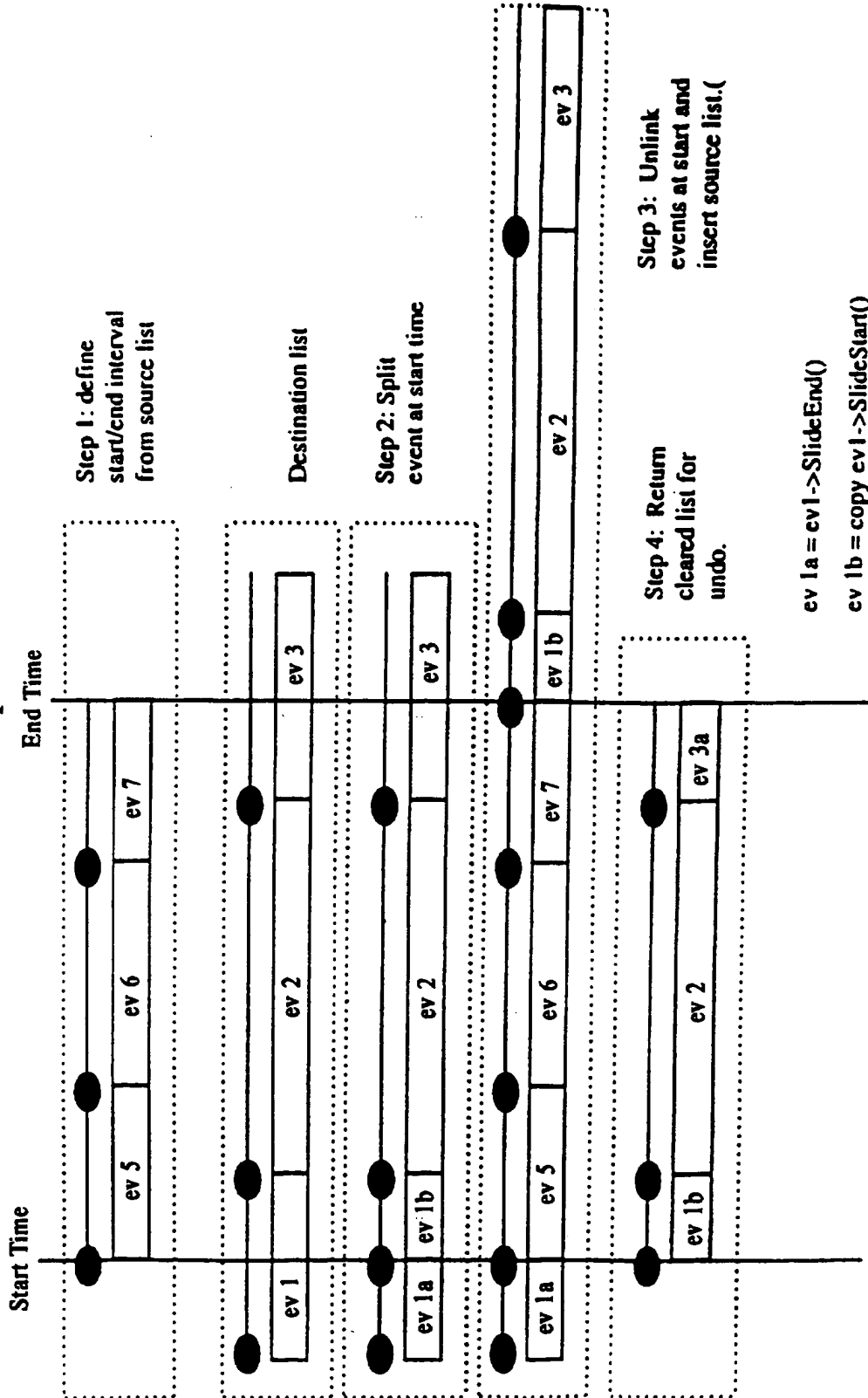


FIG. 5C

Cut Edit Operation

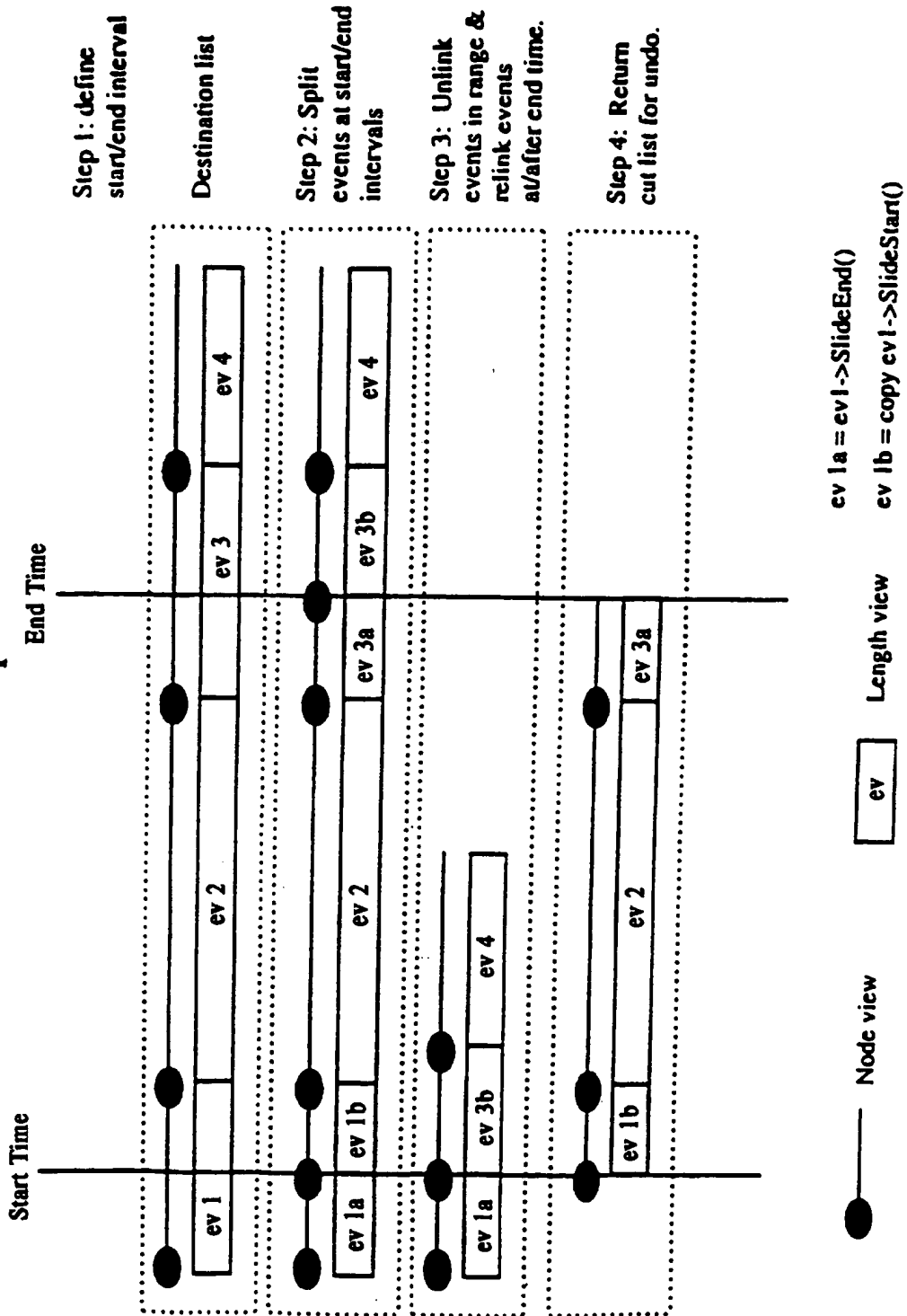


FIG. 5D

Copy Edit Operation

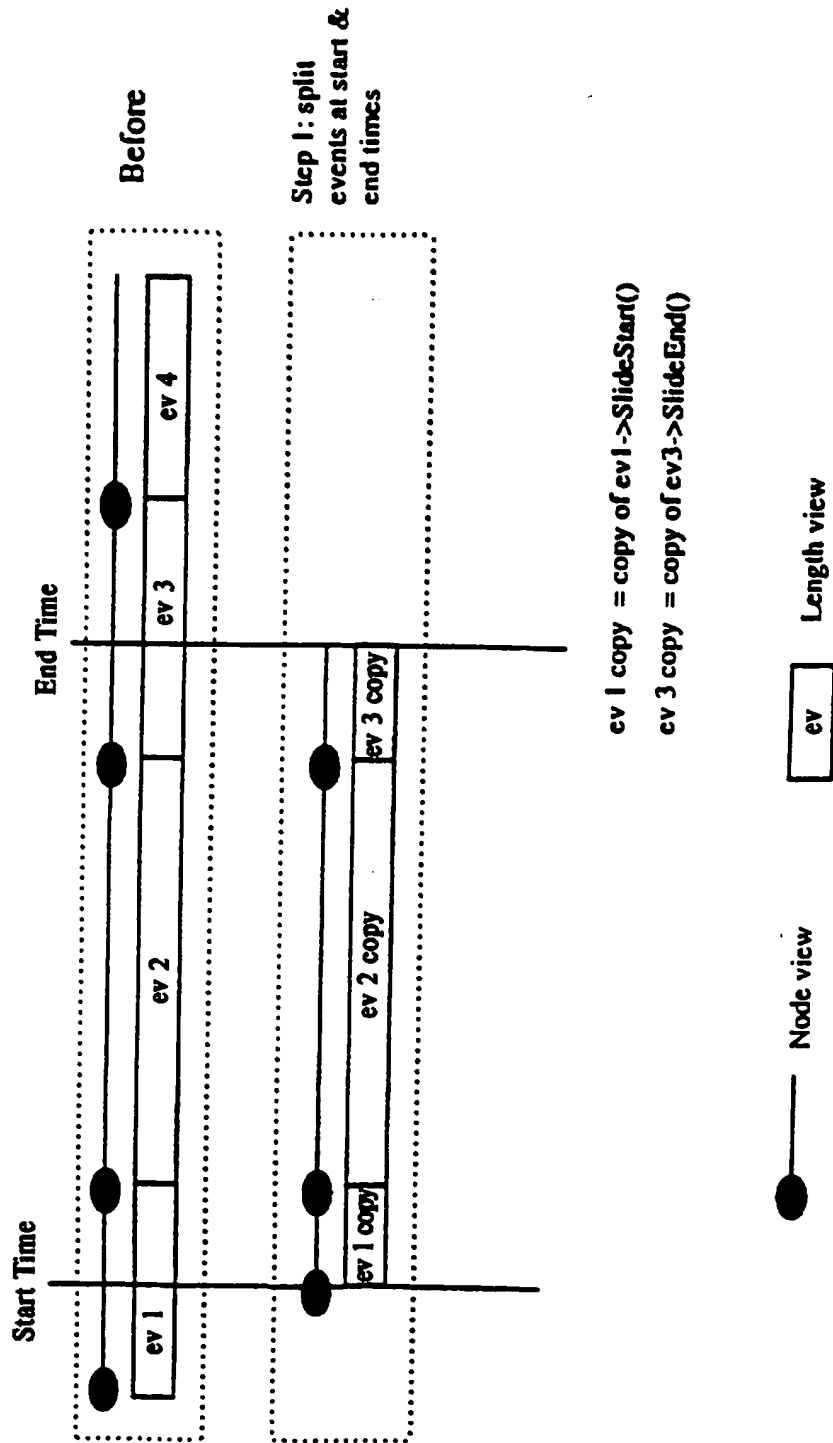
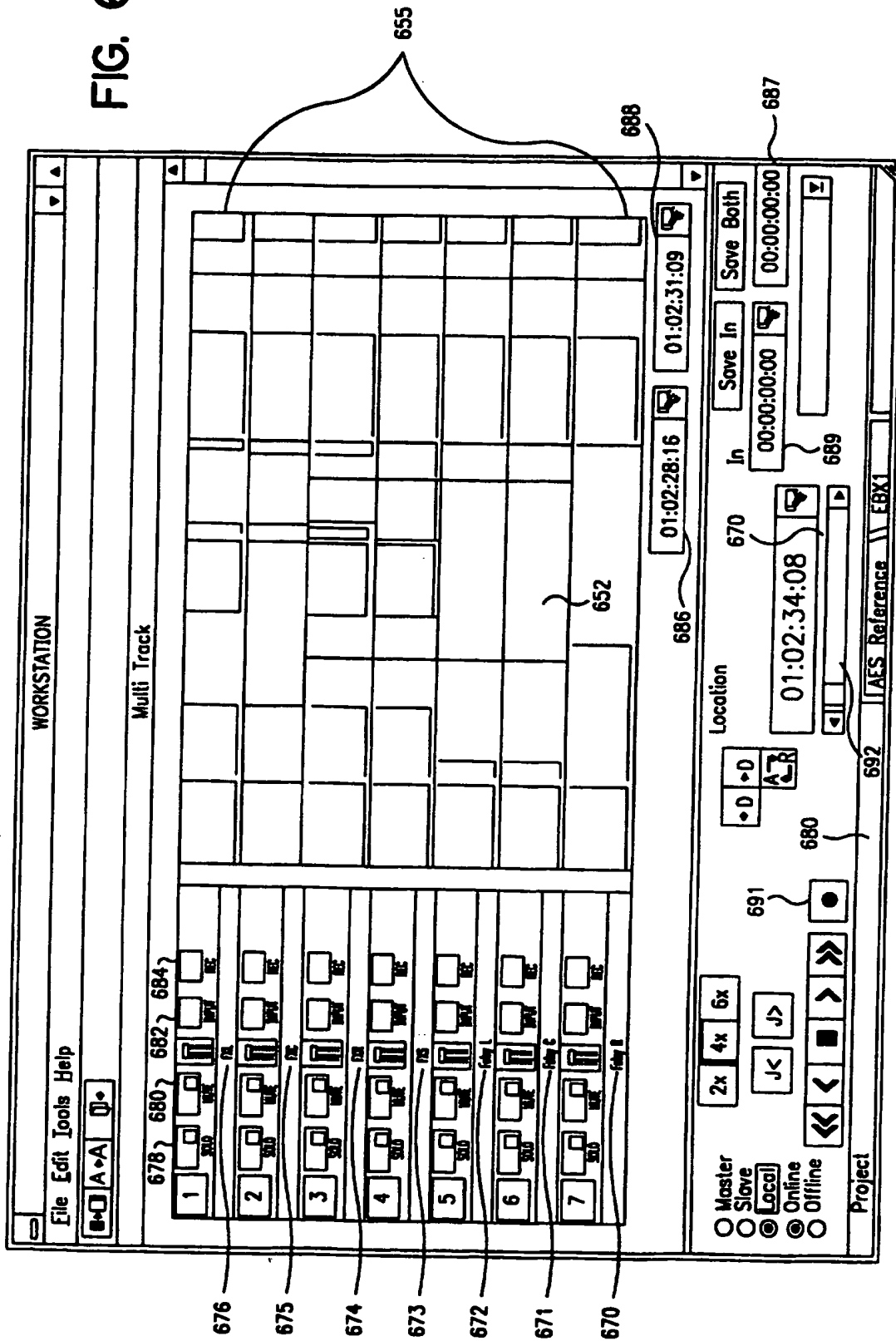


FIG. 5E

FIG. 6



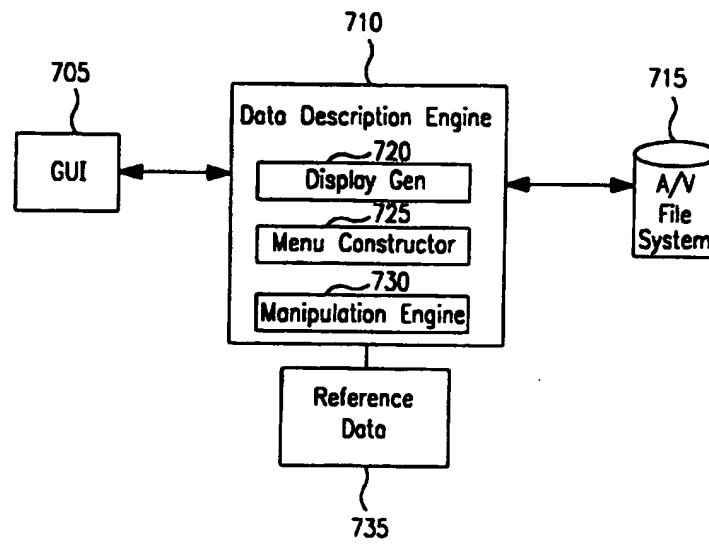


FIG. 7

FIG. 8

Project Edit Tools Help 816 OPR

812 814 818 820 810

Recorders

Track Name	Track ID	Track Name	Track ID	S	I	S	I
TR 1	15.1	TR 1	15.1				
TR 2	15.2	TR 2	15.2				
TR 3	15.3	TR 3	15.3				
TR 4	15.4	TR 4	15.4				

805

830 831 832

In 00:00:00:00 Out 00:00:00:00

Use Out Time ☒

Commit

Players

Track Name	Track ID	Track Name	Track ID	S	I	S	I
TR 1	15.1	TR 1	15.1				
TR 2	15.2	TR 2	15.2				
TR 3	15.3	TR 3	15.3				
TR 4	15.4	TR 4	15.4				

844 845 846

1 2 3 4 5 6 7 8

All None

836 833 834

00:00:00:00

Commit

Location 835

01:02:54:05

840

2x 4x 6x

J< J>

Master Slave Local Online Offline

842 843

00:00:00:00 00:00:00:00

Save Both

Project

Time Remaining(1 Track) 01:49.47

Track	Name	S	B	G	C
1.1	Tack 1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.2	Tack 2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.3	Tack 3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.4	Tack 4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.5	Tack 5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.6	Tack 6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.7	Tack 7	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
1.8	Tack 8	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Adjust Gain

-20dB +10dB

Correct Gain -6.00 dB

Time All Done

00:00:00:00

Master ☐ Slave ☐ Local ☒ Online ☐ Offline ☐

Location 005400+00

In 00:00:00:00 Save In 00:00:00:00 Out 00:00:00:00

Project AES Reference EBX1 Time Remaining(1 Track) 01:49.47